Ensure that Input Is Properly Canonicalized

William L. Fithen, Software Engineering Institute [vita[3]]

2005-10-03 L4 / D/P[4]

Failure to canonicalize input can introduce vulnerability. Inadvertently canonicalizing input multiple times can introduce vulnerability.

Description

Canonicalization is the process of transforming a potentially flexible data structure into one that has guaranteed characteristics. It is a frequent technique for input data validation. For example, the same input data "characters" can be encoded in many ways, ranging from 7-bit ASCII to variable-width multibyte Unicode. Before a program that accepts such input uses it, it is frequently required that the input be transformed into some canonical form that is universal (in the context of the program). Otherwise, even simple text comparisons (e.g., length, equal, ordering) cannot be made.

For extensive coverage of this issue see [Howard 02[5] Chapter 10: All Input Is Evil![6]].

Failure to Canonicalize (When It Was Needed)

When input with identical semantics can be supplied in multiple syntaxes, then it is usually wise to define one of the syntaxes as "canonical" and transform all of the other representations into that one before using the input. Even better is to disallow all input that is not canonical [Hoglund 04[7]].[8]

Redundant Canonicalization (Which Is Not Idempotent)

When canonicalization of input is required, be sure that it only occurs once.[9] In many representations, it is not safe to canonicalize already canonicalized input [VU#580299[10]].

References

[Hoglund 04]	Hoglund, Greg & McGraw, Gary. *Exploiting Software: How to Break Code.* Boston, MA: Addison-Wesley, 2004.
[Howard 02]	Howard, Michael & LeBlanc, David. *Writing Secure Code*. 2nd. Redmond, WA: Microsoft Press, 2002.
[VU#580299]	MacInnis, Ken. *Vulnerability Note VU#580299: Microsoft Internet Explorer contains URL decoding cross-domain vulnerability*. June 14, 2005. http://www.kb.cert.org/vuls/id/580299.

[3] http://buildsecurityin.us-cert.gov/bsi/about_us/authors/320-BSI.html (Fithen, William L.)
[5] #dsy331-BSI_Howard-02
[6] http://www.microsoft.com/mspress/books/sampchap/5957.asp#SampleChapter
[7] https://buildsecurityin.preview.us-cert.gov/daisy/bsi/articles/knowledge/guidelines/331/edit/39405b5e8457707b4d313e3d382d1e806873303d/part-article-body#Hoglund-04
[8] This can sometime be difficult when the input data is some other system's input or output and that system defines multiple representations as being legal. Examples include: DNS hostnames, IP addresses, path names in filesystems, and even numerical values.
[9] It is possible to define canonical forms that can be redundantly canonicalized without damage. For example, removing leading zeroes from integer values or removing "../" and "./" sequences from filesystem path names can be performed multiple times without harm.
[10] #dsy331-BSI_VU-580299

Carnegie Mellon Copyright

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

[1] mailto:permission@sei.cmu.edu